EVOLVE

Version 5.0i.a

USER'S MANUAL

Part II User's Guide

August 1999

EVOLVE5.0i.a is a topography-relevant process simulator. Modules exist to simulate low pressure deposition and etch (*EVLP*), high pressure deposition and etch (*EVHIP*), and thermal thin film flow (*EVFLOW*).

This user's manual for EVOLVE, Version 5.0i.a was updated in April of 1999 by Max Bloomfield and David Richards of the Process Modeling and Simulation research group at Rensselaer Polytechnic Institute.

EVOLVE5.0i.a was developed by Timothy S. Cale, Max Bloomfield, David Richards, and Sofiane Soukane, starting from previous versions. It is based on extensive work by several people, including members of the Process Simulation Laboratory in the Center for Solid State Electronics Research at Arizona State University (through 1997) and many colleagues in industry.

For more information on the history of and the contributors to EVOLVE, please see *The History of EVOLVE* section of Part II of this manual.

Users can expect to find unclear, inconsistent and even incorrect information in this manual, as it is not completed, commercial grade documentation. For example, the documentation for EVHP and EVFLOW are not as well developed as that for EVLP.

EVOLVE is under development, and we would be pleased to hear from users about potential bugs, suggestions, or novel uses for EVOLVE

Table of Contents

Introduction **4**

Input Files **6**

Output Files **8**

Guidelines **9**

Input File Format: chooser.in **12**

Input File Format: evolve.in **13**

Input File Format: hip.in **47**

Input File Format: hpcode.in **50**

Input File Format: reflow.in **53**

Input File Format : parameter.in **56**

Input File Format: move.in **58**

Input File Format: riemann.in **60**

Input File Format: specular.in **61**

Input File Format: sputter.in **63**

Input File Format: materials.in **65**

Installation Guide **66**

How to execute EVOLVE **73**

Frequently asked questions and answers **74**

The History of EVOLVE **78**

Introduction

EVOLVE is written in a combination of FORTRAN 77 and C. The version covered by this User's Guide is designed primarily for interactive execution in UNIX and X-windows environments, although batch execution is an option. Double precision is adopted for real variables used throughout EVOLVE.

This Users' Guide is based on experience with DEC (Alpha), Sun and HP workstations with at least 64 Mb of RAM (this is required for double precision installation), and should be valid for other common workstations (compilers). This document describes how to build data files for deposition, etch, and flow simulations and provides guidelines on learning to use EVOLVE5.0i as a stand-alone program. Figure 1 shows a flow diagram of EVOLVE's program structure.

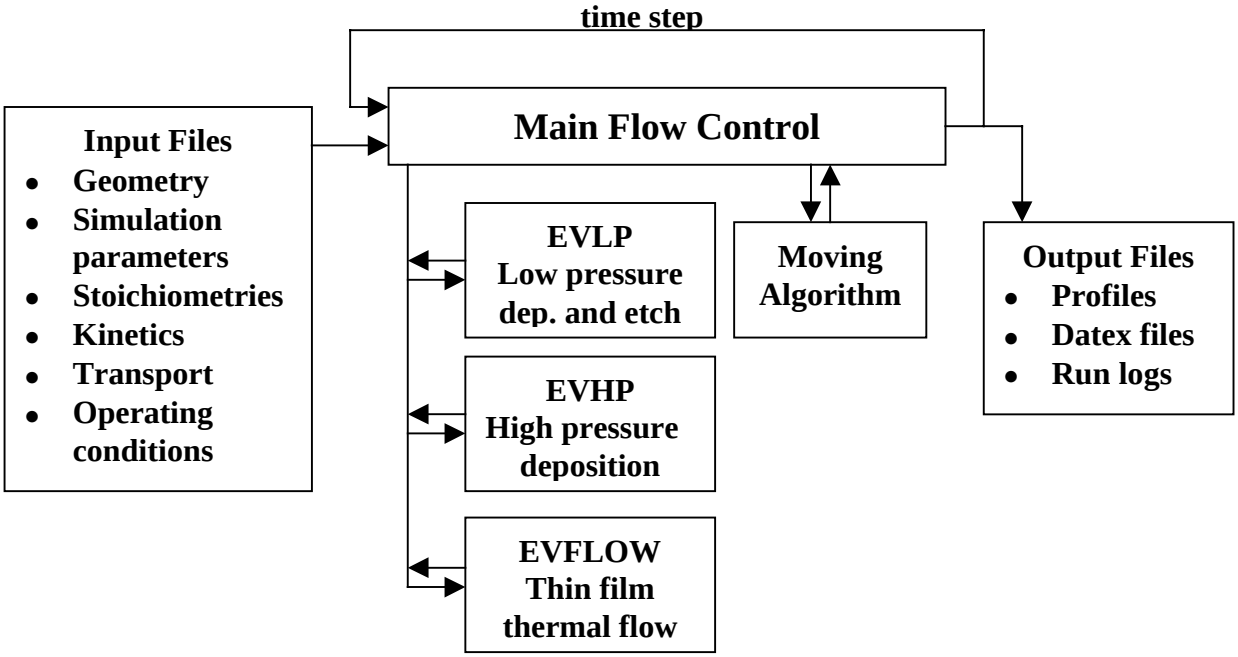

Figure 1: Program stucture of EVOLVE 5.0i.a

For "simple" deposition simulations in single trenches, typical elapsed time per incremental deposition (a step in the simulation) is usually a few seconds on a DEC alpha 600-5/266

workstation. Total simulation times for these simple processes are usually less than one minute, but depend on the input parameters. For more complex or high precision simulations, elapsed time can easily exceed several minutes. The longest simulations run to date, which involve hundreds of surface nodes, several dependent variables (species fluxes and surface concentrations - thousands of equations) and complex reaction kinetics (surface mechanisms) have taken several hours on a Sun Ultra 60 dual-processor workstation.

For simulations of deposition processes in trenches, execution time increases dramatically with increasing number of features involved in the simulation, because of the number of nodes required to represent the surface. Whenever possible, you should simulate deposition in single trenches. Simulations of etch, high pressure CVD or reflow processes take considerably longer than low pressure CVD process due to the requirements of smaller simulation time steps (in case of etch), or the nature of finite element algorithm (in cases of reflow and high pressure deposition).

Run time is affected by several parameters in the input data file, and this aspect of data file construction should be given special consideration after learning the basics of EVOLVE. For example, the execution time for one of the examples included in Part III of this user's guide (Example 5 of LPCVD) was varied by a factor of more than five by changing the values of input parameters *tol* and *relax* somewhat. The final film profile was not affected.

One very important point to remember is that the value used for *tol* (the tolerance used in the nonlinear over relaxation method (NLORM) used to determine local deposition rates, fluxes and perhaps surface concentrations) must not be too small if single precision variables are used. Even a value of 1.0D-6 has been found to be too small for simulations that include surface diffusion; *i.e.*, convergence is never attained using single precision. On the other hand, accurate solutions for processes that involve species with a wide range of fluxes and surface coverages may require tighter tolerances (1.0D-9 or even smaller).

Input Files

For the standalone version of EVOLVE, described in this guide, the input data which define the desired simulation for the program are contained in files whose names are pre-assigned in the source code. These files are:

chooser.in Controls type of EVOLVE simulation (EVLP, EVHP, or EVFLOW) and EVOLVE's interaction with other programs. Required for all simulations.

evolve.in The major input file. Defines simulation input, output, initial structure, simulation parameters, and operating conditions. For EVLP, also defines transport, reactions, and kinetics. Required for all simulations.

hip.in Contains spatial domain for gas phase homogeneous reaction, reaction kinetics, diffusivities, Only required for high pressure deposition simulations (EVHP).

hpcode.in Contains memory usage parameters for the finite element solution. Only required for high pressure deposition simulations (EVHP).

reflow.in Contains thin film physical properties such as density, viscosity, and surface tension, etc. Also contains memory usage parameters for the reflow finite element solution. Only required for thin film thermal flow simulations (EVFLOW).

parameter.in Defines memory limits and unit numbers. Required for all simulations.

move.in Controls display of output in moving algorithm solutions, and defines moving algorithm control parameters. Required for all simulations.

riemann.in Controls display of output and use of critical parameter in Riemann algorithm solutions. Required for all simulations.

materials.in Defines the material list used in EVOLVE and provides a list of some nominal starting values for each material's molar volume, surface energy and site density. Required for all simulations.

specular.in Determines the materials, species, and models used for specular reemission calculations. Required for simulations with specular species *when using the default version of the user-definable code.*

sputter.in Determines the materials, species, and models used for sputtering of solid materials. Required for simulations with sputtering species *when using the default version of the user-definable code*.

For the majority of simulations, these input files, and a file containing the initial geometry are the only input files needed; however, other files may be required by user supplied routines. For example, a file that holds the source fluxes as a function of angle might be required by a particular user-supplied *udist* subroutine.

Output Files

Output filenames for both graphical and text outputs are read from the main input file, evolve.in; however, they can be renamed interactively if desired.

The graphical output format used by EVOLVE is quite simple, consisting of sequences of ordered pairs, each pair on a line and representing a point on a plane. Sequences represent geometric regions or surfaces and are separated by two lines: a blank line and a line containing a double quotation mark followed by a label string. This simple format may be easily viewed by such utilities as XGRAPH, XMGR, XMGRACE, or in a pinch, Microsoft Excel.

The main simulation graphical output, generally named with a suffix of “grf” contains the starting structure, and *n*th intermediate surface profiles, as specified in the evolve.in file.

A second graphical output file, generally named with a suffix of “subs,” can be used to track the progress of the simulation, and is updated after moving the surface and after resizing the segments.

There are three optional graphical output files with recommended suffixes ‘prf’, ‘see’, and ‘cor’ respectively. The first file contains the final film surface in a form convenient for further simulation using EVOLVE. The second contains the last two profiles at any time during the simulation. This file is very useful if there are convergence problems during the determination of the local growth rates. The third optional file tracks the largest change in the value of the dependent variables for each iteration of the 'nonlinear over-relaxation method' (NLORM) used to determine the local growth rates.

Guidelines

The best approach to learning EVOLVE is to run the examples included in this guide. Then try different substrate profiles for the same processes. Then change the various parameters in the input file to see how they affect the film profile predictions. For example, if global kinetic expressions are being used, change the operating conditions to get a feel for their impact on sticking factors and hence film conformality or step coverage. Of course, changing the assumed kinetics (parameters in the kinetic expression) of the deposition will change the film profile predictions. Nevertheless, this may be instructional in order to get the feel of the dependence of profile evolution on a system's kinetic parameters. After running the examples provided and perhaps reading some background on the model upon which the simulator is based, you should be able to build your own data files. One basic data file is generally enough for a given process chemistry. Changes to this file can be made to simulate processes on other starting profiles or to test other kinetic expressions.

EVOLVE has several features designed to improve the accuracy of the predicted rates along the feature surfaces and hence film profiles. For example, the distance between nodes can be automatically adjusted after each step in the process in order to be about the same. This improves accuracy for given number of nodes, particularly in the presence of surface diffusion. Another feature of EVOLVE which can be used to improve the accuracy of the simulation results is that the step size is adjusted automatically based on intersections of the growth vectors.

In the low pressure code (EVLP), EVOLVE also has several features added to enhance convenience for longer simulations. Because of the large number of equations which are solved during the deposition simulation (easily several thousand), the nonlinear over relaxation method (NLORM) is used in subroutine CYCLER in which the local film deposition rates are determined. The NLORM parameter *relax* is input for use in CYCLER, in addition to the tolerance (upper limit on changes in the unknowns). *Relax* is a relaxation parameter, and is often set to one. If the method does not converge in the input maximum number of steps, you have the option of asking the program to continue with the value of *relax* the same or altered. If this occurs, then you may be using a value which is too conservative (*relax* too small) If the NLORM method "blows up", you have the option of

restarting that step in the deposition process with the value of *relax* altered. In general, lower the value and continue. With experience, this feature can be used to decrease the number of iterations required for convergence in each step of the simulation, hence decreasing the actual time lapsed for the simulation. Unfortunately, values for *relax* which speed up the simulation may also cause it to "blow up". Several simulations may be required before a good value for *relax* is determined for a "new" process being simulated. Finally, note that EVOLVE is not designed to catch all possible modes of failure in CYCLER.

As with other complex software packages, particularly those designed to be as general as EVOLVE, you may need to try several times before arriving at satisfying simulation results. The values of several of the input parameters can affect the resulting film profiles; *e.g.*, *nins* (the number of points to be inserted at an expanding corner), *isize* (whether or not to resize the segments), *fmin* (the minimum segment length kept during resizing), and *fmax* (the maximum segment length kept before resizing). Of course, it is desirable to choose values for these parameters such that changes in them do not make significant changes in the simulated film profile.

Troubles that occur during the simulation can often be tracked down using the graphical output files. Many graphing programs provides a zoom feature that can be very useful. In addition to the 'main' graph output file (suffix ‘grf’), other graph files can be used to evaluate the progress during execution to help refine the simulation. The ‘see’ file contains the last two surface profiles predicted by EVOLVE. The ‘subs’ file contains the current structure and is rewritten after reading in the profile, after moving the surface, and after resizing the segments. In addition to the output files designed for graphics, the text-based output to the screen and to the ‘out’ file may help you refine the simulation. The *iws* and *iwf* flags defined in *evolve.in* control the level detail of the text-based output. Note that text output files can be very long.

In addition to allowing user-defined substrate profiles as input, EVOLVE can generate a single, symmetric trapezoidal cross-section of an infinite trench or circular via as a starting profile for convenience. These internally generated features have flat regions on either side. Control of feature creation is discussed in Input File Format.

This version of EVOLVE assumes that the region is bounded by flat surfaces with no gradients in species fluxes or adsorbed species concentrations along the surface. Note that

this assumption is in effect whether or not the starting substrate profile is generated by EVOLVE and should be kept in mind for all substrate profiles. If a simulation yields an evolving film surface which is not parallel to these flat areas, towards their ends which are farthest from the feature, then rerun the simulation with an input profile which includes wider 'flats'. This situation is most likely to occur in the presence of surface diffusion. These flats must be oriented such that their normal vectors point from the wafer into the source volume.

A good way of testing how well you understand EVOLVE (EVLP) is to run a very low sticking factor deposition simulation in a trench with vertical sidewalls. Choose a simplistic, constant sticking factor simulation with a single reactant and single reaction in a trench of unity aspect ratio and input a sticking factor of 0.0001. Using input parameter values designed for high accuracy, you should be able to generate a very narrow void in the trench. Check the deposition rate profile closely to make sure that deposition rate does not actually increase down towards the bottom of the feature! This can occur if *spaces* or *tol* is too large. As a feature is about to close, the NLORM used in CYCLER can fail to converge, because EVOLVE is attempting to deposit in an ***extremely*** high aspect ratio feature. This situation will be obvious during the simulation, because no reasonable choices for the value of the relaxation parameter will lead to stability. The reasons for this instability are well defined, but are beyond the scope of this document. In this type of situation one may make good use of the graph file which tracks the last two profiles. If instability occurs, look into the corresponding graph file. If a feature is very close to closure, then back up a time step and increase that process step slightly. Alternatively, you might select the option that EVOLVE gives, to terminate and plot the current profile.

One of the intended uses for EVOLVE is for comparisons of predicted and experimental film profiles; therefore, starting profiles will often be digitized representations of actual substrate profiles. It is often desirable to rotate, translate and scale the profile that results from the digitization process, in case the scales and the orientations of the reference coordinate axes were not selected precisely enough. Rotation should only be done to orient the wafer surface such that it is "horizontal" and with coordinates increasing to the right, along the x axis. Thus the normal to the wafer surface, away from features, points in the y direction (into the source volume).

Input File Format: chooser.in

This input data file controls the type of EVOLVE simulation to run and EVOLVE's interaction with other programs. It is composed of a single line containing three integer values separated by white space. Those values and their names in EVOLVE are as follows:

islave The flag that indicates whether EVOLVE is being called by another program. The possible values are 0 for stand alone mode, or 1for slave mode.

ichoic The flag that indicates which module or pair of modules to use for simulation. The values of *ichoic* may be 1 for use of EVLP, 2 for use of EVHIP, 3 for use of EVFLOW, 4 for use of both EVLP and EVFLOW, or 5 for use of both EVHIP and EVFLOW.

idum This flag is reserved for future use. Any integer value will suffice.

Input File Format: evolve.in

The primary input file, ***evolve.in,*** is composed of *Blocks*, each of which is made up of one or more *Data Items*. Each individual block is associated with a single FORTRAN 77 list-directed read statement. Thus, white space (defined as blanks, tabs, or newline characters not enclosed in single quotes) is used as the Data Item delimiter. Data Items may be of three different types: real numbers, integers, and character strings

A Block may in general cover more than one line; however, this must be consistent with the read statements. In FORTRAN, if a read statement expects more Data Items than are in a given line, the next line (or part of it) is read. In general, each read statement will start reading the next line. It is important to ensure that the variable types and the number of data entries are consistent with those required by the Block. In general, each Data Item in each Block must be present in the input file, even if it not used in a particular simulation.

In the following description, *b.d* defines the *d*-th Data Item number for the *b*-th Block of the data file; *e.g.*, the fifth Data Item for Block 13 is Item 13.5. The type of Data item is indicated by a single letter enclosed in parentheses. (R) indicates a real number, (I) indicates an integer and (C) indicates a character string. Variable names used in the program are presented in *italics*.

Data Items of type (R) may be expressed in floating-point or scientific notation. Floating-point numbers are composed of an optional leading sign, a string of digits, and an optional (but recommended) decimal point. Scientific numbers are composed of the following: an optional leading sign (+,-), a numeric mantissa greater than or equal to 1.0 and less than 10.0 including a mandatory decimal point, an E or D character, an optional sign, and an integer order of magnitude. Upper and lower limits on the order of magnitude are machine dependent.

Data Items of type (I) are made up of an optional leading sign and a string of digits.

Data Items of type (C) are enclosed by single quotes and may not exceed twenty-five characters in length, or the string will be truncated.

If an error or an end-of-file condition is encountered while loading the parameters, the program prints a diagnostic message and stops execution. If you are unable to pinpoint the problem, try using a value of 1 or greater for *iws* (Data Item 5.1) and rerun the program. The Block that was being loaded when the error occurred will be displayed. The problem may be in a previous Block, because of a read statement 'overrunning' a record. The values which are read in for each Block are also echoed to the text output file, for a value of *iwf* (item 5.2) of one or more. Scanning the text output file is also an effective way of pinpointing the problem.

The program assigns numbers to reactant species, product phases and atoms in the solid products based on the order in which their names are read from the input file. Before beginning data entry, decide which reactant will be species one. This species is treated as a reference species in the program and must exist in the source volume above the wafer; *i.e.*, do not use a product species as species one, unless that species exists in the source above the wafer. You may limit the list of gas phase species to those which are involved in or influence the reaction rates in some way. Reactions are also numbered in the order in which they are input. Materials are also numbered in the order they are entered, or are indicated by the reactions which form solid phase products of which they are composed.

HEADER BLOCKS

The first 14 blocks contain filenames, the date and values related to output files. Character Data Items are limited to twenty-five characters in length and should be enclosed in single quotes.

BLOCK 1

1.1- string containing the name of the run (*inname*) (C)

This may be any string of 25 character or less. It will be written to the screen and throughout the text output file. It may be used to identify the simulation.

BLOCK 2

2.1- a reference date for the run (*date*) (I)

This may be any integer and traditionally represents a reference date that will be written to the screen and to the text output file.

BLOCK 3

3.1- the version of EVOLVE for which the data file was developed (*ivdata*) (I)

If this number doesn't agree with the value set in subroutine LOADER, then a warning message is displayed. You may continue with execution; however, be careful and check the file format. The reference value for EVOLVE5.0i.a is *042699*.

BLOCK 4

Holds four integers that control the execution of the program.

4.1 - flag to indicate if the simulation is to run in batch mode (*ibatch*) (I)

Set to zero if interactive simulations are desired (normal). Set to 1 if the simulation is to be in batch mode. Values for *ipause*, *isloop*, *ipltfl*, *ipltdv*, *ipltft*, *newnam* (BLOCK 6) and *iride* (ITEM 23.7) are reset to 0 if *ibatch* is set to 1. If *ibatch* is set to 1, and any user defined subroutines require input, then they override the value of *ibatch* - without redefining *ibatch* for the rest of the simulation.

4.2 - flag that tells EVOLVE how far to proceed (*inout*) (I)

This is useful in situations where EVOLVE is being called by another program; *e.g.*, a reactor scale simulator. For normal operation, set it to three. To read in the data files and compute the reference rates, set it to one. To compute the solution, but stop before moving the surface, set it to two.

4.3 - flag to indicates how much information about EVOLVE to print *(iabout)* (I)

The standard value of this flag is 0, but values greater than 0 will cause EVOLVE to print an extended version of its banner page to standard output before beginning the simulation.

4.4 - flag which indicates which moving algorithm to use (*lvlset*) (I)

EVOLVE5.0i.a only supports a value of 0. Other values are reserved for future expansion. This flag must be present.

BLOCK 5

Contains a set of integers that help control the input and output of the program.

5.1 - flag to indicate write level to the screen (*iws*) (I)

Set to -1 to get only step number printed to the screen. Set to 0 to get only the summary at each step after simulation starts (step number, film thickness and process time). Set to 1 to track the progress of the simulation through each step. Set to 2 to get more details of the simulation printed to the screen, particularly details associated with moving and resizing the profile. Values greater than 2 may cause printing of information not relevant to users-level use of EVOLVE.

5.2 - flag to indicate write level to the text output file (*iwf*) (I)

See description for *iws* (ITEM 5.1). *iwf* is to the output text file what *iws* is to the screen output.

5.3 - flag to create the final film profile output (*iprf*) (I)

Set to one if the final profile is needed; *e.g.*, if it is to be used as the input profile for a subsequent simulation. Set to 0 otherwise. The name for this file is read in BLOCK 9, and it can be renamed interactively if *newnam* (5.6) is set to 1.

5.4 - flag to indicate if you want the last two surface profiles written to a file (*isee*) (I)

Set to 1 to have them written to an XGRAPH compatible file. Set to 0 otherwise. The name for this file is read in BLOCK 10, and it can be renamed interactively if *newnam* (5.6) is set to 1.

5.5 - flag to indicate if you want to track the maximum correction in CYCLER (*icorr*) (I)

Set to 1 to have the absolute value of the maximum correction (over all variables) in CYCLER written to an XGRAPH compatible file. Set to 0 otherwise. The name for this file is read in BLOCK 11, and it can be renamed interactively if *newnam* (5.6) is set to 1.

5.6 - flag to indicate if you want to be prompted for filenames (*newnam*) (I)

Enter 0 if the filenames and date in the input file are to be used. Enter 1 if EVOLVE should ask you at execution whether or not you would like to enter filenames from the keyboard. If *ibatch* is one, then *newnam* is set to zero.

BLOCK 6

Contains flags that indicate the level of user interaction during execution.

6.1 - flag to determine if EVOLVE should pause (*ipause*) (I)

Set to 0 for no pauses. Set to 1 to pause at the start of the simulation in order to check the deposition rates and simulation setup. Increase to 2 to also pause after establishing new operating conditions (for a transient simulation) by a call to SETUP , the subroutine that sets up the initial time step and film thickness increments for the simulation. Further increase to 3 to additionally pause after moving the profile during each time step and after resizing the surface profile. The method of restarting after a pause is compiler dependent; therefore, don't use this feature unless you know

the restart keyword. It is usually 'go', 'continue', or just a carriage return. If *ibatch* is one, then *ipause* is set to zero.

6.2 - flag to show the trial surface in subroutine XSECTR (in MOVE_CODE) before and after treating loops and overlaps (*isloop*) (I)

Usually set to 0. Set to 1 to output the trial surface to the 'subs' file before and after dealing with any loops, which may be caused by too large a time step or moving surfaces overlapping. Although a front-tracking algorithm is used to move the surface, this does not guarantee that loops do not form should too large a time step be taken or non-adjacent sections of the surface meet, such as at feature closure. If *ibatch* is one, then *isloop* is set to zero.

6.3 - flag to indicate if flux distributions are to be put into files (*ipltfl*) (I)

Set to 1 if you want to generate files that may be used by a graphing package to display the flux distributions as functions of angle expressed in radians. If flux distributions are to be written to files, EVOLVE will ask you to enter a filename for each of the files. Be careful not to overwrite existing files by mistake. If *ibatch* is one, then *ipltfl* is set to zero.

6.4 - flag to indicate if dependent variables are to be put into files (*ipltdv*) (I)

Set to one if you want to generate files which can be used by a graphing package to display the obtained solution for a variety of dependent variables as a function of arc length. The variables available for output are reaction rates, species rates, deposition rate, species fluxes, sticking factors, atomic fractions, surface fractions, curvature, and curvature driven diffusion rates. If dependent variable values are to be written to files, EVOLVE will ask you to enter a filename for each of the files. Be careful not to overwrite existing files by mistake. If *ibatch* is one, then *ipltdv* is set to zero.

6.5 - flag to indicate if information about the final film profile is to be output (*ipltft*) (I)

Set to 1 if you want to generate files that can be used by a graphing package to display final film profile information (such as thickness) after the simulation is completed. If this information is to be written to files, EVOLVE will ask you to enter a filename for each of the files. Be careful not to overwrite existing files by mistake. EVOLVE will then ask whether the thickness information is to be determined using segment midpoints or endpoints. If *ibatch* is one, then *ipltft* is set to zero.

BLOCK 7

Contains the name of the output file to be used for text output. A suffix such as "out" is recommended. Depending upon the value entered for *iwf* (ITEM 5.2), this file can contain the information needed to trace progress of the simulation; *e.g.*, many details from the solution at each time step (flux distributions, surface fractions, local evolution rates and surface geometric information). This can help track down problems with the input file. Depending upon the value of *iwf*, a large amount of information regarding the simulation is written to this file during execution, including perhaps warning messages. These files can be extremely useful for identifying problems with a simulation, as well as to interpret the simulation results. These text output files can also be large. A filename must be present.

BLOCK 8

Contains the name of the graphical output file. This contains the starting profile as well as all surface profiles generated up to the current profile. A suffix such as '.grf' is recommended.

BLOCK 9

Contains the name of the output file to be used for the final film profile. A name must be here, even if the final film profile is not desired. A suffix such as '.prf' is recommended.

BLOCK 10

Contains the name of the output file to be used for the file that contains the last two profiles at any point in the execution (after the first step of course). A default extension such as '.see' is recommended.

BLOCK 11

Contains the name of the output file that tracks the logarithm (base 10) of the maximum absolute value of the corrections in the dependent variable values during the iterative solution procedure used in SOLVER_CODE at each time step. This is an XGRAPH compatible file. A suffix such as 'cor' is recommended.

BLOCK 12

Contains the name of an XGRAPH compatible file that contains the current structure. This file is opened and closed each time it is written to; *i.e.*, the entire structure is rewritten.

BLOCK 13

Contains the name of the file that holds the final DATEX 'geometry' file.

BLOCK 14

Contains the name of the file that holds the final DATEX 'simulation' file.

CONTROL BLOCKS

Blocks 15 to 24 contain program control parameters.

BLOCK 15

Contains 4 items related to the starting substrate:

15.1 - flag which indicates how the starting substrate is to be loaded (*inprf*) (I)

Set *inprf* to 0 to have EVOLVE generate an initial surface and single material substrate. The resulting structure is symmetric, with characteristics defined by entries in BLOCK 16.

Set *inprf* to 1 to read in a surface from a file and generate a single-material substrate. The file must contain a sequence of ordered pairs, each pair consisting of two real numbers separated by white space. Pairs must come one to a line, with no skipped lines and the sequence being terminated by the pair 10000. 10000.

Set *inprf* to 2 to have the substrate read in from a DATEX file.

15.2 - Name of the input file for the initial structure

This is used for *inprf* (15.1) of 1 or 2. A filename must be present.

15.3 - Material name for the substrate material.

This is used for *inprf* (15.1) of 0 or 1. If *inprf* is 2, the material names are read in from the DATEX file. A material name must be present. Material names are installation dependent, because they can be defined by the user. The list of materials in the standard distribution is given in Appendix A.

BLOCK 16

Contains 7 required items related to the generation of features:

16.1 - a flag indicating whether or not the feature is symmetric (*isym*) (I)

If *isym* is set to 1, EVOLVE will reflect any feature input from file about a vertical axis through the rightmost input point. Setting *isym* allows EVOLVE to take advantage of symmetry during calculations. Although *isym* may be set to 0 even for

generated features, which must be symmetrical due to their manner of specification, EVOLVE considers this likely to be an error and will attempt to warn the user.

16.2 - a flag indicating whether or not the feature has circular cross sections (*icyl*) (I)

Set *icyl*=1 for a feature of circular cross section. If *icyl*=1, then *isym* will be 1. May be used with any type of profile input.

16.3 - auto-generated feature width (*width*) (R)

The width in microns at the mouth of the feature that EVOLVE should generate if *inprf* (15.1) is set to 0.

16.4 - auto-generated feature height (*height*) (R)

The height in microns at the center of the feature that EVOLVE should generate if *inprf* (15.1) is set to 0.

16.5 - auto-generated feature sidewall angle (*wangle*) (R)

The angle in radians formed relative to the wafer surface of the feature that EVOLVE should generate if *inprf* (15.1) is set to 0. For features of rectangular vertical cross section, use $\pi/2$. For tapered features, *wangle* is less than $\pi/2$. For reentrant features, wangle is greater than $\pi/2$.

16.6 – auto- generated feature flat length (*flats*) (R)

The length in microns of the flats to either side of the feature EVOLVE should generate if *inprf* (15.1) is set to 0. This should be set to provide a reasonable surface length. After the simulation is complete, make sure that the ends of the surface profile are horizontal. If they are not, then rerun the simulator with a larger value for *flats*.

BLOCK 17

Contains five items regarding the manipulation of the input profile, if *inprf* is 1. These items must be present in any case. These are particularly useful when using digitized input profiles.

17.1- the displacement distance along the x coordinate (*deltx*) (R)

Set to a positive value, in microns, to reduce the values of the x (horizontal) coordinates read in the next BLOCK. Remember x runs left to right across the screen or page.

17.2 - the displacement distance along the y coordinate (*delty*) (R)

Set to a positive value, in microns, to reduce the values of the y (vertical) coordinates read in the next BLOCK. Remember y is negative into the substrate.

17.3 – the angle in radians to rotate the input profile in the plane of the paper (*rotate*) (R)

A positive angle rotates the input profile in the counterclockwise direction. The angle of rotation entered should make the 'flat' surface of the wafer and the x axis collinear. This feature is intended to allow you to correct for a small error in the angle coordinate axes introduced during digitization of a substrate profile from an SEM micrograph. It is important to rotate the initial structure so that the flats on either side of the feature (representing the nominal wafer surface) are 'horizontal' or parallel to the x axis.

17.4 - the scaling factor for the input profile along the x (horizontal) coordinate (*scalex*) (R)

Set to a value greater than one to expand the profile; *i.e.*, the coordinates are multiplied by this factor. Both this term and the next are intended to allow you to correct for errors introduced into the scales of the coordinate axes during digitization of a substrate profile from an SEM micrograph.

17.5 - the scaling factor for the input profile along the y (vertical) coordinate (*scaley*) (R)

Set to a value greater than one to expand the profile; *i.e.*, the coordinates are multiplied by this factor.

BLOCK 18

Contains information about the kind of process being simulated.

18.1 - flag that controls the manner in which process time is determined (*iprocs*) (I)

Set to 0 to process until a given change in position of the wafer's surface is realized on the flats (*target*, item 18.3), but stopping when a feature has closed. Set to 1 to process until a given change in position of the wafer's surface is realized on the flats. Feature closure(s) will not stop the simulation. Set to 2 to process for a particular time (*timax*, item 18.4). For values of *iproces* from 0 to 2, process conditions are constant and *nmax* (item 19.2) is checked. A value of zero or one for *iprocs* is a good choice when a problem's kinetic expressions are not well known, before running for a given time.

18.2 - parameter indicating if the process is PVD (*ipvd*) (I)

Set to 1 for PVD processes to speed up simulation, or if all of the sticking factors are unity; *i.e.*, there is no re-emission. The transmission probabilities from surface to surface are not calculated, which can speed up the simulation considerably. For CVD processes or PVD processes with subunity sticking factors, set to 0.

18.3 - the target film thickness (*target*) (R)

Used for *iprocs* equal to 0 or 1, otherwise it is a dummy variable. It is the desired change in position (y coordinate on the flat area of the wafer), in microns. This is very useful in comparing simulation results to SEM micrographs.

18.4 - the maximum process time (*timax*) (R)

Used for *iprocs*=2, otherwise it is a dummy variable. It is the duration of the deposition process that is being simulated in seconds. The use of *iproces=2* requires good kinetic information in order to be useful.

BLOCK 19

Contains 3 items that control the numerical relations used in EVOLVE:

19.1 - the nominal number of steps in the process (*nstep*) (I)

The 'nominal' time step will be set in an attempt to make the total number of simulation steps be this number. For simulations in which the endpoint is determined by a distance traveled by the flats (*iprocs* (18.1) set to 0 or 1), the predicted amount of time required for the simulation is determined by the instantaneous velocity of the surface on the flats and the remaining distance to cover. For 'timed' simulation, that is with *iprocs* set to 2, the process time remaining to reach the target time *timax* (18.4) will be used in the calculation of the nominal time step.

19.2 - the maximum number of steps to be taken (*nmax*) (I)

This is useful to over-ride input nominal number of steps (*nstep*) in order to obtain a fractional feature fill and/or to protect against too many steps being taken. It should not be greater than *nploto·npm* (see ITEM 19.3 below for *nploto; npm* is the maximum number of plots, as set in *parameters.in*). The maximum number of profiles to be plotted (*npm*) is currently set at 100. *nmax* is usually much smaller than that maximum; *e.g.*, when used to set the number of steps the simulator takes.

19.3 - the number of steps between plots (*nploto*) (I)

The output profiles are plotted at this interval of steps and at simulation termination.

BLOCK 20

Contains flags for segment resizing:

20.1 - flag to indicate the type of segment resizing which occurs (*isize*) (I)

Set to 0 for no resizing except as a result of the film movement. Set to 1 for resizing based on *spaces* (ITEM 21.1), *fmin* (ITEM 21.2) and *fmax* (ITEM 21.3), which are fractions of the nominal segment length. Typically set to 1.

20.2 - the value of isize to be used for profile setup (*isize0*) (I)

Set to 0 if the starting profile contains the entire desired surface profile. Set to 1 if you are entering only the "turning points" of the profile and/or desire resizing. The input values of *fmin* and *fmax0* (ITEM 21.4) is used. Typically set to 1.

20.3 - the step in the deposition process in which resizing begins (*istpsz*) (I)

Usually set to 1. Resizing of the initial feature is flagged using *isize0* (ITEM 20.2).

20.4 - a flag to indicate if *spaces* is to be reduced at points of high curvature (*idiv*) (I)

Set to 1 to have more nodes at points of high positive curvature on the input profile. Set to 2 to have more nodes at points of high curvature (both positive and negative curvature). Positive curvature means that the local surface is convex (from the solid side) and will expand upon deposition. Set to zero for a curvature independent value of *spaces*.

20.5 - a flag to indicate if *spaces* is to be reduced at points of high curvature on the input profile (*idiv0*) (I)

Set to 1 to have more nodes at points of high positive curvature on the input profile. Set to 2 to have more nodes at points of high curvature (both positive and negative curvature). Set to zero for a curvature independent value of *spaces*. If only the

turning points of a profile are read in, set this value to 0, because there are usually are not enough nodes present to provide good values of curvature.

20.6 - a flag to control surface dekinking (*ikink*) (I)

EVOLVE has two dekinking criteria. Set ikink to 1 to turn on Independent-Angle dekinking, to 2 to turn on Total-Angle dekinking, to 0 to turn off both types of dekinking, and to 3 to turn on both types of dekinking. (See *fkink*, Data Item 21.5 for an explanation of the dekinking criteria.)

BLOCK 21

Contains information about segment resizing.

21.1 - the nominal segment length (*spaces*) (R)

The nominal segment length, *spaces* (in microns) should be chosen small enough to represent the details of the profile that is being deposited onto. If *spaces* is chosen too small, then too many points will be generated. This depends on the total length of the profile. Note that if *isize0*=1 is chosen to resize the initial profile from the entered turning points, then there could be more nodes generated than indicated by simply dividing the length of the surface by *spaces*.

Since the computer execution time increases with the number of segments, the number should be kept as small as possible while providing acceptable answers. To see if the solution depends on the number of initial segments, increase or decrease the value of *spaces* somewhat after getting the solution for one value of *spaces*. For smooth, micron-scale features use 0.05 or 0.1 μm to start, except for large features; *e.g.*, see example EX.3.WF6.

21.2 - the minimum fraction of the nominal segment length (*fmin*) (R)

The fraction of *spaces* which is considered small enough to combine segments; *i.e.*, segments shorter than this fraction of *spaces* will be combined with a neighboring segment. A good starting value is 0.2.

21.3 - the maximum fraction of the nominal segment length (*fmax*) (R)

The fraction of *spaces* that is considered large enough to divide a segment. Nodes are inserted between the nodes that define the long segment. A good starting value is 1.5, but decrease to 1.2 for smoother curves.

21.4 - the maximum fraction of the nominal segment length for initial profile(*fmax0*) (R)

The fraction of *spaces* that is considered large enough to divide a segment on the input profile. Nodes are inserted between the nodes which define the long segment. A good starting value is 1.5, but decrease to 1.2 for smoother curves.

21.5 - the critical value for determining whether three segments form a kink (*fkink*) (R)

In Independent-Angle dekinking, a sequence of three segments which form two angles of opposite sign, both with absolute values equal to or greater than *fkink*·•/2, will be considered as forming a kink. For Total-Angle dekinking, a sequence of three segments that form two angles of opposite sign, the sum of the absolute values of which is equal to or greater in than *fkink*·π, will be considered as forming a kink. Note that in both cases, smaller values of *fkink* find more kinks. If a kink is found, then the middle segment of the three is removed if it is shorter that *spaces*. Otherwise nothing is done.

BLOCK 22

Contains information about how the profile is to be moved.

22.1 - the minimum fraction of the nominal time step which is to be taken (*tfmin*) (R)

A good value to start with is 0.5. For more accurate simulations close to feature closure, use a smaller value. The nominal time step used in EVOLVE, dictated by the rate of surface evolution and the input target or process time, can be changed in MOVE_CODE because growth vectors overlap or cross region element boundaries, if *icheck* (ITEM 22.5) is greater than 0.

22.2 – controls the insertion of points at 'expanding' corners (*nins*) (I)

This is the number of points per $\pi/2$ difference in segment angles to insert at an expanding corner. Preferably set to an even number. Try two to start with. This number can influence the shape of profiles somewhat, particularly in high sticking factor depositions (such as PVD).

22.3 - the minimum number of nodes for which an overlap is indicated (*ncover*) (I)

In subroutine XSECTR in MOVE_CODE, intersections between surface segments are determined. If a intersection involves segments which are close to each other, then the resulting overlap is treated as a loop to be trimmed. If the intersection involves segments whose indices differ by more than *ncover*, then the loop is treated as an overlap (the feature is closing). The number to be entered depends on the feature depths and the length of the segments (*spaces*). As a rough guide, set *ncover* to the shallowest feature depth divided by *spaces* (this feature should also be one which may form a void).

22.4 - flag indicating if the deposited film is to be 'blended' with the underlying films, if it is the same material (*iblend*) (I)

If *iblend* is 1, then material deposited on top of the same material will be converted to the same region. Usually set to 1. If iblend is set to zero, then a number of region

elements can be generated, and the maximum number will often be exceeded. The maximum number (*nregm*) is set in *parameters.in*.

22.5 - flag which indicates how the time step should be adjusted if the growth vectors intersect each other or cross material boundaries (*icheck*) (I)

Intersections of growth vectors with other vectors or material boundaries are checked for as specified by the value of icheck, as listed below. If the fraction of the nominal time step required to prevent these intersections is below *tfmin* then it is reset to *tfmin*. The latter logic can lead to loops, but a reasonable value of *tfmin* is sometimes required to get results in a reasonable number of time steps and will not affect the profile too much. EVOLVE always calls bcheck to check for intersections with material boundariies. Set icheck to the following to check for intersections with:
1,3,5,7 (gcross) other growth vectors
2,3,6,7 (tcross) the trial surface
4,5,6,7 (ccross) the unmoved surface (coord)
icheck=1 call gcross
icheck=2 call tcross
icheck=3 call gcross and tcross
icheck=4 call ccross
icheck=5 call gcross and ccross
icheck=6 call tcross and ccross
icheck=7 call gcross, tcross and ccross

BLOCK 23

Contains 7 items related to the nonlinear solvers used in determining 1) the reference rates (Levenberg-Marquardt or LM) and 2) the local reaction rates (nonlinear-over relaxation method or NLORM):

23.1- the maximum number of iterations for convergence in the LM algorithm (*maxitr*) (I)

This value is only important for simulations that include surface species. A value of 100 usually suffices. If convergence is not obtained in *maxitr* iterations, then the program gives you the opportunity to enter in an incremental number of iterations and the program continues.

23.2 - the tolerance on the surface fractions determined in the LM algorithm (*tolr*) (R)

This value is only important for simulations that include surface species. A good starting value is 1.0D-6. It may be necessary to tighten the tolerance (decrease *tolr*) for simulations that involve widely different fluxes to the wafer or reaction rates on the wafer. In any case, the tolerance should be set small enough so that there are no observable differences in the answers upon decreasing it even more.

23.3 - controls which nonlinear equation solver to use to determine local rate (*ismeth*) (I)

EVOLVE5.0i.a only supports a value of 1, which indicates the non-linear over-relaxation method, NLORM.

23.4 - the maximum number of iterations for convergence in the algorithm used to determine local rates (*maxit*) (I)

A value of 10000 usually suffices for the NLORM. If convergence is not obtained in *maxit* iterations, then the program gives you the opportunity to enter in an incremental number of iterations and the program continues. This is often a good idea, because the convergence can be very slow for the NLORM, because of the value of *relax* used to determine the step size of the independent variables. The value of *relax* needed for stability may lead to very slow convergence.

23.5 - the tolerance on the fluxes and surface fractions determined in the algorithm used to determine local rates (*tol*) (R)

A good starting value is 1.0D-6. It may be necessary to tighten the tolerance (decrease *tol*) for simulations that involve widely different fluxes to the wafer or reaction rates on the wafer. In any case, the tolerance should be set small enough so that there are no observable differences in the answers upon decreasing it even more, keeping in mind the limits of resolution of SEM micrographs. Execution time increases significantly as *tol* decreases, so use the loosest (highest) value you can. Check to make sure that the predicted profile does not depend on *tol* for your application; however, the differences in predicted profiles between a tolerance of 1.0D-5 and a tolerance of 1.0D-7 are typically in the thousandths of microns.

23.6 - the relaxation parameter used in the NLORM algorithm (*relax*) (R)

Set to 1.0 for new problems, but change to 1.25 to achieve faster convergence for stable problems. If you change it to 1.25 and the problem 'blows up', reset to 1.0. In general, lower the value if the problem fails to converge. If the problem fails to converge for values of *relax* less than 0.1, then something is probably wrong in the problem setup. If convergence is not obtained, then EVOLVE allows you to change the value of *relax*.

23.7 - indicates whether or not you can change the parameters of the algorithm used to solve for the local rates (*iride*) (I)

Set to zero in order to be able to change the relaxation and scaling parameters of the solution algorithm used to determine the local rates, if the algorithm either fails to converge or 'blows up'. Set to one to let the program terminate. Usually set to zero, in order to use values of the relaxation and scaling parameters that will allow for relatively rapid convergence. Such values also tend to lead to instability. Thus, the ability to enter new values is critical, particularly if you are most of the way through a simulation. If *ibatch* (ITEM 4.1) is one, then *iride* is set to one.

BLOCK 24

24.1 - the uncertainty in angles to be used when comparing angles (*perta*) (R)

A good value to use is 1.0D-6 radians.

24.2 - the uncertainty in the cosine of angles (*pertc*) (R)

A good value to use is 1.0D-6.

24.3 - the uncertainty in lengths (*perts*) (R)

A good value to use is 1.0D-6 microns.

24.4 - the minimum magnitude to be considered nonzero (*eps*)

A good value to use is 1.0D-12 for double precision installations.

CHEMISTRY BLOCKS

Blocks 25 through 32 define the chemistry of the process. Note that gas phase products that do not affect the reaction rates, through the reaction rate expressions, do not have to be tracked. You should plan the input data file so that the components and reactions are in the order you wish and are internally consistent; *e.g.*, use the same species as reactant number one throughout the data file.

BLOCK 25

Contains 6 items which specify the numbers and types of species:

25.1 - the number of gas phase (ballistic) reactants (*ncomp(1)*) (I)

This is the number of gas phase species in the system that participate in the reactions; *i.e*, there is no need to include inerts or products which do not influence the reactions rates. Inert species or non-reacting product species may be tracked if desired, unless *ipvd* (ITEM 18.2) is one.

25.2 - the number of chemisorbed surface species (*ncomp(2)*) (I)

This is the number of species considered to be chemisorbed on the surface. The sum of the surface concentrations of these species and the vacant site concentration must add to the total site concentration (*sitec*).

25.3 - the number of physisorbed species (*ncomp(3)*) (I)

The chemisorbed species and physisorbed species are tracked differently, because the sum of the surface fractions of chemisorbed species and the vacant species should add to one. No such constraint is placed on the physisorbed species.

25.4 - the number of solid phase products (*ncomp(4)*) (I)

This should be no more than the number of reactions, since each reaction is assumed to produce no more than one solid product.

25.5 - the number of solid materials present (*ncomp(5)*) (I)

A value should be entered here, and it is checked against the number of materials in the original substrate and the materials formed by the reactions-- not the solid phases formed by the reactions, but rather the materials that these solid phases make up. In many cases, these are the same, but not necessarily (see EX.4.WSI).

25.6- the number of reactions (*nreact*) (I)

This is the number of reactions that occur.

BLOCK 26

Contains the names of the ballistic species or collision species in the high-pressure case (*spname*) (C), in single quotes and up to 25 characters in length. Thus, this block contains *ncomp(1)* entries, on one or more lines. See examples.

BLOCK 27

Contains the molecular weights of the ballistic species or collision species (*wm*).(R). Thus, this block contains *ncomp(1)* entries, on one or more lines.

BLOCK 28

Contains the names of the chemisorbed species (*spname*) (C) in single quotes and up to 25 characters in length. Thus, this block contains *ncomp(2)* entries, on one or more lines. This block will not exist if there are no chemisorbed species.

BLOCK 29

Contains the names of the physisorbed species (*spname*) (C) in single quotes and up to 25 characters in length. Thus, this block contains *ncomp(3)* entries, on one or more lines. This block will not exist if there are no physisorbed species.

BLOCK 30

Contains the names of the solid phase products (*spname*)(C) in single quotes and up to 25 characters in length. Thus, this block contains *ncomp(4)* entries, on one or more lines.

BLOCK 31

Contains the molar volume (*vol*) (R) of each solid phase compound in cubic centimeters per gram mole. This is computed as the ratio of the molecular weight to the density in grams per cubic centimeter.

BLOCK 32

Contains the number of elements in the solid phase products (*nlmnts*) (I)

BLOCK 33

Contains the names of the elements in the solid phases (*elname*) (C), in single quotes and up to 25 characters long. There are *nlmnts* entries in this block, on one or more lines.

BLOCK 34

Contains the atomic stoichiometric coefficient matrix for the elements in solids phase products (*astoich*)(R). There are *ncomp(4)*·*nlmnts* items in this Block, entered carefully. There are *nlmnts* lines, each with *ncomp(4)* entries. These entries are the number of atoms of each element listed Block 33 in solid the solid phase products listed in Block 30. See the extended explanation in Part III of this manual, in the description of the tungsten silicide example **(EX.4.WSI)**.

BLOCK 35

Contains information about which materials each reaction occurs on, and which material any solid phase products become a part of (*istoic*)(I). Enter this information on two lines, each of which contains *nreact* entries. First line contains the material indices of the material that each reaction occurs on. A zero indicates that the reaction occurs on any material. The second line indicates which material any solid phase reaction products become part of. The reaction will occur on the material formed as well as the material indicated by the first line.

BLOCK 36

Holds the transpose of the generalized stoichiometric coefficient matrix (*stoich*)(R). There are *nspec=ncomp(1)+ncomp(2)+ncomp(3)+ncomp(4)* rows in this Block, each with *nreact* entries. Thus, each line contains the generalized stoichiometric coefficients of one species in all reactions. See the extended explanation in Part III of this manual, in the description of the tungsten silicide example **(EX.4.WSI)**.

KINETICS BLOCKS

Blocks 37 and 38 contain kinetic information.

BLOCK 37

Contains the integer array indicating the type of reaction rate expression used for each reaction (*irtype*) (I). This block has *nreact* entries. Enter 1 for a flux based reaction rate expression with a constant factor. This is used for simple, reactions with constant (at a given

temperature) values for the sticking coefficients and single reactants. Enter 2 for a general global (macroscopic) irreversible deposition reaction in terms of reactant pressures, such as might be observed in kinetic experiments. Enter 3 for a mechanistic sequence, which include reversible reactions and are written in terms of fluxes and surface fractions. Surface diffusion can be included for *irtype*=3. Rate expression types 4 and 5 are similar to irtype=3, except rates are written in terms of partial pressures or concentrations.

If object or source code is available, users may code their own rate expressions into the subroutine *urate*, up to a maximum of *nrusrm* (as set in *parameters.in*) user rate expressions. The *irtype* of any such reaction would have to be set accordingly here. See the section on user-definable code in Part III of this manual for more details.

BLOCK 38

Contains *nreact* (25.6) lines with each line corresponding to one reaction, in the order given by the stoichiometric matrix from Block 36. The items on each line depend on the relevant value of *irtype* set in the previous block. In the following formulae, *rparam(i,j)* indicates the *i*th entry of the *j*th line of this block.

If *irtype* is set to 1, the reaction rate may depend on the local flux of each species. As usually used, the rate is written as a single component reaction with constant sticking factor. For convenience, the reaction rate form is written more generally:

$$rate(j) = rparam(j,1) \cdot \exp\left(-\frac{rparam(j,2)}{temp}\right) \prod_{i=1}^{ncomp(1)} flux(i)^{rparam(j,i+2)}$$

The pre-exponential factor of the flux-based reaction (in practice, usually the sticking factor) is given as the first entry on the line in corresponding to the reaction with *irtype* set to 1. Usually for an *irtype* of 1, *nreact* is 1 and the reaction depends on the flux of a single reactant with a sticking factor between 0 and 1; however, *irtype* of 1 is always a valid reaction type and the rate may depend on any or all species fluxes. Each line corresponding to a reaction with an *irtype* of 1 will contain *ncomp(1)*+2 items.

If *irtype*=2 a global, multicomponent reaction rate expression, which can be expressed in the following fairly general form, is assumed. The general form of the rate expression for *irtype*=2 is

$$rate(j) = \frac{rparam(j,1)\exp\left(-\frac{rparam(j,2)}{temp}\right)\prod_{k=1}^{ncomp(1)} press(k)^{rparam(j,k+2)}}{\left(1+\sum_{k=1}^{ncomp(1)} term(j,k)\right)^{rparam(j,4\cdot rncomp(1)+3)}}$$

where

$$term(j,k) = rparam(j,3+ncomp(1)+3(k-1))\cdot \exp\left(-\frac{rparam(j,4\cdot ncomp(1)3(k-1))}{temp}\right)\cdot press(k)^{rparam(j,5+ncomp(1)+3(k-1))}$$

If *irtype* is 2 for a given reaction, then the rate parameters for the reaction are entered on one or more lines. As a check when finished, count the number of entries on the line to make sure it has 4·*ncomp(1)*+3 entries. Refer to the general rate form given above to guide you through the items on the line; *i.e.,* for each reaction:

1. The pre-exponential factor of the rate expression. This is the constant in front of the reaction kinetic expression. The units for this factor must be consistent with the pressure dependencies such that the deposition reaction rate has units of gmoles/cm^2/sec.

2. The activation energy for the reaction in units of Kelvin (actually enter E/R_g). If a given reaction is not activated, enter 0.0.

3. The next *ncomp(1)* positions contain the reaction orders with respect to each component.

4. The parameters in the denominator of the rate expression are entered as a group of three for each component. The adsorption constant, the heat of adsorption and the pressure dependence. Thus there are 3·*ncomp(1)* entries here for each reactant. For a constant adsorption constant, enter 0.0 for the heat of adsorption.

5. The exponent on the denominator. Enter 1.0 even if all terms in 4 above are 0.0.

If *irtype* is set to 3 then *nreact* (25.6) multicomponent mechanistic reactions from the following fairly general class are assumed. Fluxes (gmole/cm^2/sec) are represented by *F*, surface concentrations (gmole/cm^2) are represented by *S*, *nd=ncomp(1)+ncomp(2)* and *nm=nd+ncomp(3)*. The multiplications over the chemisorbed surface fractions (*nc2*) start at zero because that is the vacant site concentration. The general form of the rate expression for an *irtype* of 3 is

$$rate(j) =$$

$$rparam(j,1)\,temp^{rparam(j,2)} \exp\left(-\frac{rparam(j,3)}{temp}\right) \prod_{k=1}^{ncomp(1)} F(k)^{rparam(j,k+3)} \cdot$$

$$\prod_{k=0}^{ncomp(2)} S(k)^{rparam(j,k+ncomp(1)+4)} \prod_{k=1}^{ncomp(3)} S(k+ncomp(2))^{rparam(j,k+nd+4)} -$$

$$rparam(j,nm+5)\,temp^{rparam(j,nm+6)} \exp\left(-\frac{rparam(j,nm+7)}{temp}\right) \prod_{k=1}^{ncomp(1)} F(k)^{rparam(j,nm+7+k)}$$

$$\prod_{k=0}^{ncomp(2)} S(k)^{rparam(j,8+nm+ncomp(1)+k)} \prod_{k=1}^{ncomp(3)} S(k+ncomp(2))^{rparam(j,8+nm+nd+k)}$$

As a check when finished, count the number of entries on each line. They all should have 2(*nm*)+7 entries. Refer to the general rate form given above to guide you through the items on each line (reaction):

1. The pre-exponential factor of the forward rate expression. This is the constant in front of the forward kinetic expression. The units for this factor must be consistent with the flux and surface coverage dependencies such that the deposition rate has units of gmoles/cm^2/s.

2. Rate expressions often feature temperature raised to some power. The power is entered. Enter 0.0 for temperature independence.

3. The activation energy for the forward reaction in units of Kelvin (actually enter E/R_g). If a given reaction is not activated, enter 0.0.

4. The next *ncomp(1)* positions contain the reaction order with respect to the flux of each component.

5. The next *ncomp(2)* positions are the dependencies of the forward rate on surface fractions of chemisorbed species.

6. The next *ncomp(3)* positions hold the dependencies of the forward rate on the surface fractions of physisorbed species.

7. The above 5 steps are repeated for the reverse reaction.

8. The total site density, upon which the reaction is based, is then added as the final parameter.

If irtype(i)=4, then a mechanistic expression will be used, written in terms of concentrations and surface concentrations. For irtype(i)=5, a mechanistic expression will be used, written in terms of pressures and surface concentrations.

TRANSPORT BLOCKS

Blocks 39 through 42 contain information on the flux distributions for the ballistic species. Blocks 43 and 44 contain information on surface diffusion.

BLOCK 39

Contains the type of flux distribution model used (*ipdf*) (I) for each of the *ncomp(1)* ballistic species.

BLOCK 40

Contains the flux distribution parameters (*fparam*) (R) for each of the *ncomp(1)* ballistic species. This block will consist of *ncomp(1)* lines, the *i*th line with *nfp(i)* real number values that characterize the source flux of the *i*th ballistic species, with *nfp(i)* determined by the corresponding value of *ipdf* (Block 39). In the following formulae, *fparam(i,j)* denotes the *j*th entry of the *i*th row (or the *j*th parameter of the flux of the *i*th ballistic species).

For *ipdf* set to 0, 1 or 2, a single parameter characterizes the flux distribution. For *ipdf* equal to or greater than 3, the number of parameters for the model, *nfp*(*ipdf*), is entered through *udist*, a user-defined subroutine. (See the section on user-definable code is Part III of this manual.) The default version of *udist* is an example of how to use UDIST to load flux data from a file.

For *ipdf* set to 0, this is the cosine distribution, and is appropriate for CVD processes, (x is in

$$g(x) = \cos^{fparam(i,1)-1}(x)$$

radians):

For *ipdf* set to 1, *fparam* (i,1) is the parameter of the generalized cosine distribution

$$g(x) = \left[fparam(i,1)\sin^2(x) + \cos^2(x)\right]^{-1}$$

If *fparam* (i,1) is greater than 1, then the distribution is narrower than the cosine and there is 'beaming'. If *fparam* (i,1) is less than 1, then the distribution is wider than the cosine distribution. This is often used to model fluxes in sputter deposition. To obtain cosine distributions, set *fparam* =1.0, or use an *ipdf* of 0.

For *ipdf* set to 2, *fparam* (i,1) is the parameter (standard deviation) of the exponential distribution (x is in radians):

$$g(x) = \left[2\pi \cdot \mathit{fparam}(i,1) \cdot \exp\left(\frac{-x^2}{2\,\mathit{fparam}^2(i,1)} \right) \right]^{-1/2}$$

This is often used to model ion fluxes in a plasma.

For *ipdf* equal to 3, the flux distribution information is stored in a file whose name is requested during execution. (This overrides *ibatch.*) The information should be given as the relative flux at specific angles from the normal (theta), and represents g(x) above in tabular form. Start with theta of zero and continue through π/2. EVOLVE will stop reading the file upon reading a theta value greater than π/2.

For *ipdf*>3, a user-defined distribution function is used. It is defined in subroutine *udist*. The number of parameters can be up to *nfpm*, which is set in *parameters.in*. This can only be done if you have source or object code. See the section on user-definable code in Part III of this manual for more details.

BLOCK 41

An array of flags which indicate if each species can re-emit specularly (*ispclr*) (I). This block consists of one line with single integer entries for each of the *ncomp(1)* ballistic species. Set an entry to 0 if a species should not be modeled specularly. The models used to describe the conversion of specular to diffuse species must be supplied by the user in subroutine *ubeta*. See **Example 11** in Part III of this manual for an extended explanation of specular transport models..

BLOCK 42

Contains 6 parameters used in the integrations of the flux distribution functions.

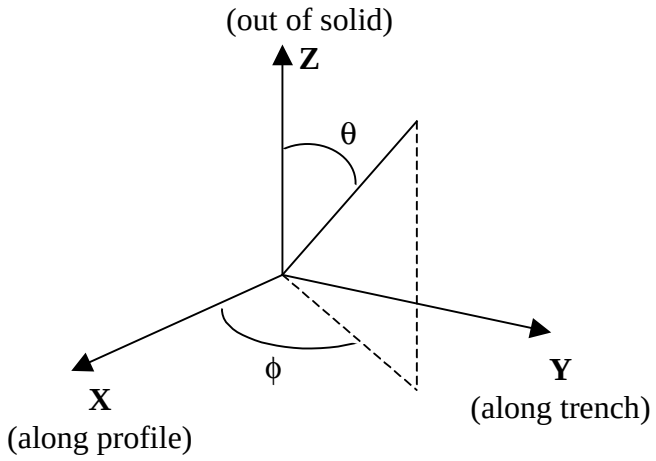

Figure 2 : EVOLVE's coordinate system

42.1 - the number of intervals about the z-axis used in azimuthial integrations (*nphi*) (I)

EVOLVE uses a local coordinate system with the z-axis pointing out of the solid. *nphi* is the number of angular increments in the domain of the azimuth angle, ϕ, from zero to π, for the purposes of integration. The maximum value for *nphi* (*nphim*) is set in *parameters.in*. Usually, a value of 20 or 30 is satisfactory.

42.2 - the number of angular intervals measured from the z-axis to the plane tangent to the surface (the x-y plane) used in zenith integrations (*ntht*) (I)

ntht is the number of angular increments in the domain of the zenith (polar) angle θ, from zero to $\pi/2$. The maximum value for *ntht* (*nthtm*) is set in *parameters.in*. Usually, a value of 20 or 30 is satisfactory; however, for flux distributions which are sharply peaked, a higher value is recommended.

42.3 - the number of intervals used in integrations over surfaces (*ng*) (I)

This is the number of steps used to integrate over surfaces to obtain the differential transmission probabilities in infinite trenches. The maximum value for *ng* (*ngm*) is set in *parameters.in*. Usually, a value of 10 is satisfactory.

42.4 - the maximum number of reflections for specular species (*nrflct*) (I)

This is the maximum number of reflections through which a specular species may be tracked. Moieties that reflect more than this number without being converted to reactive flux are lost to the simulation, unless the user deals with them in the user-definable routine *ubeta*. A value of *nrflct* of 20 allows the vast majority of moieties to be tracked to reactive conversion or exit of the feature. For moieties which enter at a very high angle (from the normal to the wafer surface, much larger values may be needed, depending upon the specular to reactive conversion models used. The maximum value for *nrflct* (*ntrfltm*) is set in *parameters.in*.

42.5 - a flag indicating whether to test the flux computations for an inert, diffusely emitting species (*notest*) (I)

BLOCK 43

Contains two parameters associated with surface diffusion.

43.1 - flag to indicate the driving force for surface diffusion (*idiff*) (I)

Set to 0 for no surface diffusion, independent of other values. Set to 1 for surface concentration driven surface diffusion. Set to 2 for curvature driven surface diffusion. Set to 3 for combined concentration and curvature driven surface diffusion. In order to use concentration driven surface diffusion, your model must include surface species.

43.2 - minimum diffusivity (*dmin*) (R)

If the surface diffusivities entered are lower than this value, then surface diffusion is ignored. In general use 0.0, since 1.0D-12 is added to the value in EVOLVE before comparing with values of the surface diffusivities of each surface species. Units are cm^2/sec.

BLOCK 44

Contains a number of lines, each containing a pair of surface diffusion parameters (*dparam*). The number of diffusivity parameter pairs is determined by the value of *idiff* (ITEM 43.1). For *idiff*=0, no values are required. For *idiff*=1, then there are *nsurf*=[*ncomp*(2)+*ncomp*(3)]·*ncomp*(5) pairs; *i.e.*, a pair for each surface species on each material. The order in which the parameters pairs are read corresponds to the species order in *ncomp*(2) and *ncomp*(3); *i.e.*, the first *ncomp*(5) pairs correspond to the first chemisorbed species. For *idiff*=2, there *ncomp*(5) pairs, representing the curvature driven surface diffusivity of each material. For *idiff*=3, there are (1+*nsurf*)·*ncomp*(5) pairs. The first *ncomp(5)* pairs correspond to the curvature driven diffusivity.

The first entry in each pair is the pre-exponential factor for a diffusivity. The second item in each pair is the activation energy for diffusion (actually enter E/R_g for units of Kelvin). If a constant value is desired, enter that constant for the preexponential factor and zero for the activation energy. These diffusivities are in cm^2/sec.

OPERATING BLOCKS

Blocks 45 through 47 contain information about the operating conditions.

BLOCK 45

Contains the temperature, in Kelvin(*temp*) (R) and pressure, in torr (*tpress*) (R).

BLOCK 46

Contains flags for each of the *ncomp*(1) ballistic species indicating how the operating conditions for the deposition are entered (*ioper*)(I).

Set *ioper* to 1 for pressure in torr, set to 2 for concentrations in gmole/cm^3 and set to 3 for fluxes in gmole/cm^2/sec. These operating condition flags are entered in the order of their species numbers, which were assigned when the species information was read into the program.

BLOCK 47

The values for the operating conditions (*oper*)(R). After being read by the program, in any form, the complementary values are computed; *e.g.*, if the pressure is entered, the concentration and flux are computed. The relationships used are those for the equivalent ideal gas; if the flux of a very directional species is specified on input, the partial pressure of a Maxwellian gas which provides that same flux is computed.

Input File Format: hip.in

This input file provides additional parameters for high-pressure deposition process simulation.

BLOCK 1

1.1 The height of the solution domain (*uhigh*) (R).

This value should be selected large enough in order that it does not have significant effects on the simulated film conformality. Three microns is a good point to start.

1.2 The relaxation factor used in iteration procedure (*alpha*) (R).

This parameter affects the coverage of the program somewhat. Smaller values of *alpha* lead to better coverage, but use more iteration steps. EVOLVE takes the input value to begin the simulation. If the equations do not converge after each hundred iterations, EVOLVE will reduce the value of *alpha* by a factor of four. The maximum number of the iteration is set to be five hundred.

BLOCK 2

The total pressure of the deposition system considered (*ptotal*) (R)

This parameter is used in estimating the species diffusivities in the gas phase. If the user selects to input species binary diffusivities instead of calculating the effective diffusivities in the mixture, this entry is dummy.

BLOCK 3

3.1 The number of the inert carrier gases in the system (*inert*) (I).

The number of the species whose concentration distributions need to be calculated is read in from evolve.in as *ncomp(1)* (Data Item 25.1). The inert carrier gases read in from here will not be calculated during the simulation, although the will be used for the effective diffusivity estimations.

BLOCK 4

The molecular weights of the inert gases (*wm*) (R). This block contains *inert* (Data Item 3.1) entires. If inert is zero, this block then does not exist.

BLOCK 5

The partial pressures of the inert gases (*press*) (R). This block contains *inert* (Data Item 3.1) entries. If inert is zero, this block then does not exist.

BLOCK 6

6.1 - The number of the gas phase reactions (*nhomrx*) (I).

BLOCK 7

The types of the reaction rate expression used for each gas phase reaction (*ihotyp*) (I). This block contains *nhomrx* (Data Item 6.1) entries. If *nhomrx* is zero, then this block does not exist. This will supercede the item of the same name entered in ***evolve.in.***

BLOCK 8

The transpose of the generalized stoichiometric coefficient matrix (*hstoi*) (I). As described in ***evolve.in***, there are *ncomp(1)* rows in this block, each with *nhomrx* (Data Item 6.1) entries. Thus, each line contains the generalized stoichiometric coefficients of one species in all reactions. If *nhomrx* is zero, then this block does not exist.

BLOCK 9

This block contains *nhomrx* (Data Item 6.1) lines. The entries in each line depend on the relevant value of *ihotyp* (Block 7). Users may refer to the ***evolve.in*** detailed description. If *nhomrx* is zero, then this block does not exist.

BLOCK 10

10.1 - A flag for the method of estimating species diffusivities in the gas phase (*idest*) (I).

If *idest* is zero, the user selects to use species binary diffusivities as the diffusivities in the mixture instead of calculating locally. The value of one indicates that diffusivity estimations are needed and the following blocks are the parameters needed for those estimations.

BLOCK 11

The contents of this block depend on the value of *idest* (Data Item 10.1).

If *idest* is zero, this block contains species binary diffusivities (*bidiff*) (R). In this case, there are *ncomp(1)* (evolve.in Data Item 25.1) entries.

If *idest* is one, this block contains characteristic Lennard-Jones lengths of the gase phase species (R). In this case there will be *ncomp(1)+inert* (evolve.in Data Item 25.1 + hip.in Data Item 3.1) entries. The order of these lengths should be those of the *ncomp(1)* gas species followed by those of the *inert* inert gases.

BLOCK 12

The characteristic energies of interaction of gas phase species (R). This block contains *ncomp(1)+inert* entries. The order of these energies should be those of the *ncomp(1)* gas species followed by those of the *inert* inert gases.

Input File Format: hpcode.in

This input file provides memory limits for the high pressure finite element routines. In an effort to make the input format user-friendly, this file has been designed to use value/keyword pairs for input. Associated with each input is a unique string of alphabetic characters that is used to distinguish it from all other inputs.

The file contains two types of lines: value/keyword pairs and comments. Value/keywords lines are distinguished by starting with numeric characters (defined as digits, decimal points, positive and negative signs.) They contain a real or integer value followed by the keyword string[1]. Comments lines may be blank or may start with any non-numeric characters.

Memory Limits

Note: All memory limits are of integer type.

Limit/Keyword: NELEMX

The maximum number of triangles available to represent the solution domain.

Limit/Keyword: IBCMX

The maximum number of boundary elements available to represent the boundaries of the solution domain.

Limit/Keyword: CODEMX

[1] On some UNIX systems, the keyword string may need to be enclosed in single quotes to be recognized.

The maximum number of points in the mesh, including T6 midpoints, available to represent the solution domain.

Limit/Keyword: LENSUF

The bounds of a work array used to trianglate the solution domain. Should be equal to or less than IBCMX.

Limit/Keyword: IBCMX

The maximum number of boundary elements available to represent the boundaries of the solution domain.

Limit/Keyword: MXT

The maximum number of triangles available to represent the solution domain.

Limit/Keyword: MXV

The maximum number of mesh vertices available to represent the solution domain.

Limit/Keyword: MXB

The maximum number of boundary elements available to represent the boundaries of the solution domain.

Limit/Keyword: MXJB

The maximum number of edges, boundary and internal, available to represent the mesh of the solution domain.

Limit/Keyword: LENW

The dimension of the working array used by the meshing algorithm. Should be at least 200 + 6(3MXV + LENSUF).

Input File Format: reflow.in

This input file provides thermophysical properties, simulation parameters, and memory limits for the viscous thermal flow process simulations. In an effort to make the input format user-friendly, this file has been designed to use value/keyword pairs for input. Associated with each input is a unique string of alphabetic characters that is used to distinguish it from all other inputs.

The file contains two types of lines: value/keyword pairs and comments. Value/keywords lines are distinguished by starting with numeric characters (defined as digits, decimal points, positive and negative signs.) They contain a real or integer value followed by the keyword string[2]. Comments lines may be blank or may start with any non-numeric characters.

Thermophysical Properties:

Property: Viscosity **Type**: Real **Keyword**: VISCOSITY

This entry specifies the dynamic viscosity of the film under the flow conditions in poise.

Property: Surface Tension **Type**: Real **Keyword**: SIGMA

This entry specifies the surface tension of the film under the flow conditions in dyne/cm.

Property: Density **Type**: Real **Keyword**: DENSITY

This entry specifies the mass density of the film under the flow conditions in g/cm^3.

Simulation Parameters

[2] On some UNIX systems, the keyword string may need to be enclosed in single quotes to be recognized.

Parameter: Reflow Material Index **Type**: Integer **Keyword**: RFINDX

This is the position in the "materials.in" file indicating the substance to be reflowed. It may correspond to any material with the exception of the "Void" material.

Parameter: Penalty Parameter, ε **Type**: Real **Keyword**: EP

This is the penalty function parameter, ε, used in the finite element algorithm. It is a small number, typically $.10^{-9} \square \varepsilon \square 10^{-4}$.

Memory Limits

Note: All memory limits are of integer type.

Limit/Keyword: NELEMX

The maximum number of triangles available to represent the film.

Limit/Keyword: IBCMX

The maximum number of boundary elements available to represent the boundaries of the film.

Limit/Keyword: CODEMX

The maximum number of points in the mesh, including T6 midpoints, available to represent the film..

Limit/Keyword: LENSUF

The bounds of a work array used to trianglate the film. Should be equal to or less than IBCMX.

Limit/Keyword: IBCMX

The maximum number of boundary elements available to represent the boundaries of the film.

Limit/Keyword: MXT

The maximum number of triangles available to represent the film.

Limit/Keyword: MXV

The maximum number of mesh vertices available to represent the film.

Limit/Keyword: MXB

The maximum number of boundary elements available to represent the boundaries of the film.

Limit/Keyword: MXJB

The maximum number of edges, boundary and internal, available to represent the mesh of the film.

Limit/Keyword: LENW

The dimension of the working array used by the meshing algorithm. Should be at least 200 + 6(3MXV + LENSUF).

Input File Format : parameter.in

```
10    nbm is the maximum number of ballistic species
10    nim is the maximum number of intermediate surface species
5     nspm is the maximum number of solid phases involved in reaction
5     nem is the maximum number of atomic species (elements) in the
solid phases
15    nrxm is the maximum number of reactions
10    nratem is the maximum number of reaction rate models
4     nfpm is the maximum number of flux distribution parameters in the
pdfs
5     npdfm is the maximum number of flux distribution models
90   mxdata is the maximum number of flux data points which can be read
in user file
4    nbpm is the maximum number of parameters in the models for
specular reemission (beta)
5     nbetam is the maximum number of models for specular reemisson
4     nypm is the maximum number of parameters in the models for
sputter yield
5     nyldm is the maximum number of models for sputter yield
551   nsmax is the maximum number of surface segments (elements).
10    ninsm is the maximum number of nodes to be inserted at expanding
corners.
10    npm is the maximum number of plots
20    intdim is the maximum number of overlaps (intersections)
100  nregm is the maximum number of regions.
30 nmtlsm is the length of the array used to store material names
20    ngm is maximum number of interation increments for geometry
6     nspdm is the maximum numer of integration increments for speed
90    nphim is the maximum number of integration increments for phi
180   nthtm is the maximum number of integration increments for theta
100   nrfltm is the maximum number of specular reflections
3     ngpmax is the dimension of parameter vector of length NP, used in
riemann_code
************************************************************
5   nstip is the unit number for the standard input (keyboard)
6   nstop is the unit number for the standard output (screen)
3   nfinp is the unit number of the input file.
4   nftop is the unit number of the text output file.
8   nfgop is the unit number of the graphical output file
9   nfpop is the unit number of the profile output file
10  nflop is the unit number of the file holding the last two profiles
11  nfcop is the unit number of the file holding the corrections from
cycler (convergence info)
12  nfdip is the unit number of the datex 'geometry' input file
13  nfdop is the unit number of the datex 'geometry' output file
14  nfsop is the unit number of the datex 'simulation' output file.
15  nfssp is the unit number of the file for the current substrate
16  nfdbp is the unit number of the file for backing up geometry data
17  nfdmp is the unit number of the file for geometry data in move
18  nfsbp is the unit number of the file for backing up sim. data
```

```
19  nfsmp is the unit number of the file for sim. data in move
21  nfuup is the unit number for user defined models & integers
22  nfdvp is the unit number of the file for writing dep. variables
23  nfrsp is the unit number of the file for writing scaled rates
24  nflow is the unit number of the file for reading reflow.in
25  nfhip is the unit number for reading EVHIP files
```

Input File Format: move.in

This input file controls display of output in moving algorithm solutions, and defines moving algorithm control parameters

BLOCK 1

The following items are ignored by EVOLVE for values less than, or equal to 0.

1.1 - Equivalent to *iwf*, but confined inside mover_code, *iwe* (I)

1.2 - Equivalent to *iws*, but confined inside mover_code, *iwm* (I)

1.3 - Equivalent to *iws*, but confined inside vector_code,*iwv* (I)

1.4 - Equivalent to *iws*, but confined inside xsector_code, *iwx* (I)

1.5 - Equivalent to *iws*, but confined inside check_code, *iwc* (I)

1.6 - Equivalent to *iws*, but confined inside mover_code and defreg_code, *iwd* (I)

1.7 - currently not used, iwu (I)

1.8 - see descriptions in **evolve.in**. *iwf* (I)

BLOCK 2

2.1 - for use in mover_code, see descriptions in **evolve.in**. *ipause* (I)

2.2 - for use in mover_code, see descriptions in **evolve.in.** *isloop* (I)

2.3 - for intermediate profile inspection. *ispave* (I) (1-generate profile graph)

2.4 - for use in mover_code, see description in **evolve.in**. *icheck* (I)

2.5 - for use in move_code, see description in **evolve.in**. *nins* (I)

BLOCK 3

3.1 - critical arc length for void formation. *arccrt* (R)

3.2 - nominal scaling factor. *sfactr* (R)

Input File Format: riemann.in

This input file controls the Riemann algorithm solutions

Block 1

1.1 - *istring*(I), a specific segment number to apply this block to. Non-positive values apply to all segments.

1.1 - iws(I) for use in riemann_code, see description in **evolve.in**.

1.2 - iwf(I) for use in riemann_code, see description in **evolve.in**.

Block 2

This block is unlike other blocks and uses the keyword system used in the **hpcode.in** and **reflow.in** files. *It is entirely optional, and for most simulations, should not be present.* It may be used to reset the value for *angcrt,* the smallest angle between adjacent segments by which the Riemann problem may divide to obtain an angular growth rate derivative.

The entire block is composed of a real value followed by white space and the string 'ANGCRT' enclosed in single quotes. This value of *angcrt*, if present, supercedes the default value of *angcrt* used within riemann_code of 0.01 radians.

This block is used almost exclusively for reflow simulations.

Input File Format: specular.in

This input file, read by the default user-definable routine *ubeta,* contains information about which model, model parameters, and material are used for specular transport. **Note:** The file format give here is valid only as long as it is not redefined in the routine *ubeta*. A particular user with object or source code who modifies *ubeta* may completely alter or eliminate the use of this file.

The following set of **5 blocks** is repeated, in order, once for each specular species.

Block 1

A string used to identify to the user the model of specular reflection EVOLVE is to employ. (C)

Block 2

The name of the specular species, as given in *evolve.in* (C)

Block 3

A string used to identify the material, as in *material.in*, from which the species is reflecting. 'All' or 'all' is reserved for species which reflect identically off of all materials. (C)

Block 4

The number of model parameters employed by this model. The number must be greater than or equal to 1. (I)

Block 5

The appropriate model parameter on one line. (R)

Block 6

The particular string ‘Done’ or ‘done’ is used to indicate that no more information about specular species is to be read. Any lines after this block are silently ignored.

Input File Format: sputter.in

This input file, read by the default user-definable routine *uyield*, contains information about which model, model parameters, and material are used for specular transport. **Note:** The file format give here is valid only as long as it is not redefined in the routine *uyield*. A particular user with object or source code who modifies *uyield* may completely alter or eliminate the use of this file.

The following set of **5 blocks** is repeated, in order, once for each sputtering species/sputtered material pair.

Block 1

A string used to identify to the user the species/material sputtering model that EVOLVE is to employ in yield calculations. (C)

Block 2

The name of the sputtering species, as given in *evolve.in* (C)

Block 3

A name of the sputtered material, as given in *material.in*, that is sputtered by the species listed in Block 2. The particular string 'All' is reserved for species which sputter all materials using the same model. (C)

Block 4

The number of model parameters employed by this model. The number must be greater than or equal to 1. (I)

Block 5

The appropriate model parameters, listed on one line. If your model does not require any parameters, set Block 4 to 1 and list a dummy parameter on this line. (R)

Block 6

The particular string ‘Done’ or ‘done’ is used to indicate that no more information about specular species is to be read. Any lines after this block are silently ignored.

Input File Format: materials.in

This data file defines the materials and is required for all simulations. It provides a list of material names and some nominal starting values for each material's molar volume, surface energy and site density. Materials may be added to the file, but the user is cautioned that a non-standard materials.in file is a potential source of errors if unknowingly used in a simulation not designed using that particular file.

```
'Silicon '            10.   1000.  2.5d-9
'Polysilicon '        10.   1000.  2.5D-9
'Silicon_Dioxide '    10.   1000.  2.5D-9
'Silicon_Nitride '    10.   1000.  2.5D-9
'Aluminum '           10.   1000.  2.5D-9
'Aluminum_Nitride '   10.   1000.  2.5D-9
'Titanium '           10.   1000.  2.5D-9
'Titanium_Nitride '   10.   1000.  2.5D-9
'Titanium_Silicide '  10.   1000.  2.5D-9
'Tungsten '           10.   1000.  2.5D-9
'Tungsten_Nitride '   10.   1000.  2.5D-9
'Tungsten_Silicide '  10.   1000.  2.5D-9
'Titanium-Tungsten '  10.   1000.  2.5D-9
'Ti-W-N '             10.   1000.  2.5D-9
'Copper '             10.   1000.  2.5D-9
'Molybdenum'          10.   1000.  2.5D-9
'Metal '              10.   1000.  2.5D-9
'Dielectric '         10.   1000.  2.5D-9
'Resist '             10.   1000.  2.5D-9
'Substrate '          10.   1000.  2.5D-9
'Deposit '            10.   1000.  2.5D-9
'Void '               10.   1000.  2.5D-9
'Unknown  '           10.   1000.  2.5D-9
'   '                 10.   1000.  2.5D-9
```

Installation Guide

Requirements:

Every effort has been taken to make EVOLVE5.0i.a conform to ANSI FORTRAN 77 and ANSI C standards. The motivation for this has been to make the code as portable as possible. It has been successfully compiled and run in many different UNIX environments. To the best of our knowledge, there are no issues that outright prevent the code from being compiled and run successfully on any architecture for which ANSI FORTRAN and ANSI C compilers are available.

This installation guide is written for implementation on UNIX workstations. EVOLVE has specifically been written to allow its integration with other programs; *i.e.*, the routines which simulate dep/etch/reflow processes can be used as a standalone simulator or as a subroutine called by another program (*e.g.*, SAMPLE). This Installation Guide is for the standard distribution, for which EVOLVE is a standalone program. The data files required are described in the User's Guide.

The Contents of the Distribution:

The EVOLVE5.0i.a distribution is composed of three main directories: `bin`, `examples`, and `code`. `bin` will hold the binary which you will compile, `examples` holds a large number of sample simulations which are ready to be run (see Distributed Examples), and `code` holds the source code of EVOLVE5.0i.a.

EVOLVE5.0i.a is distributed in the standard tarred, gzipped format. Tools for unzipping and untarring files are widely available. For example, the program `gunzip` from the GNU Project, and the SYSTEM V UNIX utility `tar` may be used. It is assumed that an explanation of the expansion of the archive is unnecessary.

The tar file will be named something along the lines of `evolve5.0i.a.tar.gz`.

Building EVOLVE

Building EVOLVE5.0i.a requires ANSI compliant FORTRAN 77 and C compilers. Further, there compilers must be interoperable, such that object code and symbol tables created by the two compilers be compatible. Most commercial compiler suites fulfill this requirement. The GNU Project has compilers available which are (as of 4/1999) interoperable.

Configuration

Upon moving into the code directory, you will see that EVOLVE is set up to use the `make` utility. It is highly recommended that you use this utility, as compiling EVOLVE by hand may be difficult and is unsupported. The file `local_compile` has been set up as a central location for configurational changes. Included in the distribution are several versions of this file, each prepared for a different set of compilers. Available are `local_compile`s for Sun, HP, and DEC native compiler suites and for the GNU project compilers. Using one of these files as a starting point for configuration will make the process much easier.

`local_compile`

This file contains `make` definitions which you must customize to work with your compilers. These definitions are:

`CC` and `F77` which should be set to the path names to your compilers.

`FOFLAGS` and `COFLAGS` should be set to any optimization flags you wish to use in the compilation.

`FLIBS` which should be set to the F77 flags that cause linking to the appropriate libraries. The math and C libraries must be linked to, as must any libraries required as a result of other compiler flags. (*e.g.*, If optimizing is turned on above, an optimization library may require linking.)

`FUFLAGS` and `CUFLAGS` should be set to cause the F77 and C compilers to output the same format to the symbol table. See the section below: **Compiler matching**.

FDATAFLAGS and CDATAFLAGS should be set adjust the default word length and alignment rules of the compilers in the unlikely event that those of the compilers do not match each other.

FCFLAGS and CCFLAGS should be set cause the compilers to skip the linking stage. In each case, this is almost always -c. Additionally, any compiler specific flags may be set here, such as a range checking flag or an ANSI C flag.

You most likely will not need to change any of the remaining make variables, but they are included here for completeness.

FDFLAGS and CDFLAGS should be defined as the debugging flags for the compilers. These may be left blank if not compiling in debug mode.

FLOUT is the flag that directs the F77 compiler to output to a specific filename. It is most likely –o.

FLFLAGS is a catch-all for any F77 flags that the linker needs to properly do its job. It is included solely for non-standard compiler/linkers and should remain blank in almost all cases.

Compiling and linking

After you have established local_compile you may proceed to build EVOLVE by running make. The default make target should compile and link EVOLVE, placing the binary executable in the bin directory.

Troubleshooting the Build:

Common problems and their solutions:

- Cannot execute the make command.

  Verify make is in your path. Use an explicit pathname if necessary.

- `make` fails to compile any object files.

  You have no path to the compilers. Alter `local_compile` to use explicit pathnames.

  Permission are set incorrectly. Change the ownership and group status of the distribution to belong to you. Add read permissions on the code files and write permissions to the directories within code.

- `symbol not found` or `code unsatisfied` errors during linking

  Your FORTRAN and C compilers are using different conventions regarding underscores and function names in the symbol table. See the section on **Compiler matching.**

Compiler matching

This may be the most difficult part of configuring EVOLVE. Although an F77 and C compiler may produce completely interoperable binary code, they may not use the same convention to place the lookup names in the symbol table- even compilers from the same suite can do this!

Often, symbol names are simply the function or subroutine names prefixed or postfixed with one or more underscores, but this is compiler dependent. Thus, the C function `FOO(BAR)` may end up as `__FOO_` in the symbol table, but the F77 code `CALL FOO(BAR)` may try to reference `_FOO_`, depending on the compiler used.

Your job is to read the compiler documentation, looking for the flags that alter this behavior. When you find them, you must decide which to use. Use any errors returned by the linking stage to guide you. Routines with names regarding memory allocation or sizing of arrays may most often be assumed to be C routines. Also, F77 routine names are limited to six characters. For example, if part of the output from the linking stage is:

```
Unresolved reference: evolve_size_
 first encountered in evolvr.o

Unresolved reference: mover
 first encountered in mv_allocate.o
```

Here `evolve_size` and `mv_allocate` are C functions, and `evolvr` and `mover` are F77 routines. Obviously, by default, the F77 compiler postfixes an underscore, or it would not be looking for one at the end of a routine name. Also apparent is the fact that the C compiler does not add an underscore.

Thus, any flag which causes the C compiler to add an underscore, or causes the F77 compiler to not add an underscore is the appropriate one. (Don’t do both!)

Distributed Examples

EVLP examples

EX.1.PVD	PVD
EX.1C.PVD	PVD in a feature of circular cross section
EX.2.CVD	Simple CVD
EX.2C.CVD	Simple CVD in a feature of circular cross section
EX.3.WF6	LPCVD Tungsten
EX.4.WSI	LPCVD Tungsten silicide
EX.5.SURF	PVD with surface diffusion
EX.6.PECVD	PECVD
EX.7.CURVE	PVD with curvature driven surface diffusion
EX.7C.CURVE	PVD in a feature of circular cross section with curvature driven surface diffusion
EX.8.COLL	Collimated PVD
EX.8C.COLL	Collimated PVD in a feature of circular cross section
EX.9.POLY	Polysilicon deposition into high a high aspect ratio trench
EX.10.TIW	Sputter deposition of Ti-W
EX.11.SPEC	PVD with specular transport
EX.12.IPVD	IPVD sputtering of TaN
EX.13.ETCH	Anisotropic reactive ion etch

EX.13C.ETCH	Anisotropic RIE in a feature of circular cross section

EVHP examples

EX.14.HPCVD	High-pressure CVD deposition of SiO_2 from TEOS
EX.14.HPCVD	High-pressure CVD deposition of SiO_2 from TEOS with an alternate chemical model

EVFLOW examples

EX.16.FLOW	BPSG glass film reflow

How to execute EVOLVE

It is recommend that you run EVOLVE as follows:

1. Include the path to evolve/bin in your path. (Add to path statement in .login, .cshrc, or other such initialization file. Alternatively, type

   set path = ($path PATHNAME)

   at the UNIX prompt, where PATHNAME is the full path to evolve/bin.

2. Copy examples from the evolve/examples subdirectory into your file space. Keep an unchanged set of the example files that are distributed with EVOLVE in a safe place.

3. Run EVOLVE by typing **EVLP** at the UNIX prompt. In the example directory, the **runit** script may be used.

*Keep in mind that this version of EVOLVE does **not** ask for input file names unless you specify in **evolve.in**. All major *.in input filenames are preassigned inside source code. As described in the User's Guide, these *.in files are: **evolve.in, chooser.in, move.in, riemann.in, materials.in,** (and sometimes include **sputter.in, specular.in, hip.in,** and **reflow.in**). For your convenience, I have placed every set of these files in each example subdirectory*

Frequently asked questions and answers

The following Q&A were compiled by our graduate students during their learning of EVOLVE. We hope this will help those who just start to learn EVOLVE. If you have discovered something in your use of EVOLVE that you feel would be particularly beneficial to the EVOLVE user community, please contact the authors.

GENERAL QUESTIONS

Q: What is the best way to start learning EVOLVE?

A: The best way to start learning evolve is to get copies of examples and to run them 'as is' and then change the parameters to see what effect this has, but don't change more than 2 parameters at a time in case you get an error. It is best to put examples in their own folders.

Q: What files do I need in my folder for running an example?

A: The files always required are: `evolve.in`, `chooser.in`, `riemann.in`, `materials.in`, *and* `move.in`. *The files which are sometimes required are:* `specular.in` *(for simulations employing specular species),* `sputter.in` *(for simulations employing sputtered materials),* `hip.in` *(for high pressure simulations only),* `hpcode.in` *(for high pressure simulations only), and* `reflow.in` *(for reflow simulations only).*

TROUBLESHOOTING

Q: If I run into fatal errors (i.e. the program won't run), what can I do to try to get EVOLVE to run?

*A: Try changing the write level (*iwf *and* iws) *to 2. Then spend time carefully going through the header information which* EVOLVE *prints out at the start of a run. 90% of error may be found by careful review of the headers! Are the reactions/species/materials that* EVOLVE *outputs really the ones you think you put in? Are the flat-wafer calculations reasonable?*

Q: What are some common fatal errors? (Why won't my simulation even start?)

A: Some common fatal errors are:

- *TYPOS! The* EVOLVE *input format can be quite demanding, so verify everything in the header outputs. Some common typos are:*
  - *One of the large blocks of numbers, such as the stoichiometry matrix (*stoic*) or the kinetic expression parameters (*rparam*) has the wrong number of entries.*
  - *Inconsistencies in material names between* `evolve.in`*.,* `materials.in`*, and any datex file used for input. "Silicon_Dioxide" is* ***not*** *the same as "Silicon_dioxide"!*
  - *Spacing in the input datex file. Unfortunately FORTRAN 77's record-based input routines are not very forgiving, so check your datex file's format carefully against one of those included in the examples directory.*
- ncomp(1) *is not equal to the number of ballistic species for low pressure simulations.*
- ncomp(1) *is not equal to the collision species for high pressure simulations.*
- ioper *does not have the same number of components as the number of ballistic species*
- oper *does not have the same number of components as the number of ballistic species*
- *for high pressure simulations (in* `hip.in`*): binary diffusivity does not have the same number of components as the number of collision species.*
- *If an error such as* `array index out of bounds` *is produced, then chances are that some piece of code was not allocated enough memory. Check* `parameters.in` *to verify that you have enough space allocated for the corresponding model in* `evolve.in`*.*
- *An error such as* `Allocation failed for`*… indicates that your environment is limiting the memory resources you may use. The most common fix for this is to run "*`unlimit datasize`*" in your* UNIX *window before running* EVOLVE*. If this fails to help, consult your system administrator or pare down memory usage in* `parameters.in`*.*

Q: What are some common semi-fatal errors (Why can't my simulation finish?)

A: Some of more common errors that can cause a simulation to stop prematurely are:

- *The maximum number of plots* (nplotm) *or the maximum number of steps (*nmax*) has been exceeded. (Very common!)*
- *The time step is far too large, causing a growth vector to exceed the simulation domain. Try using* iproc *0 or 1 for initial time step calibration.*
- *Your surface has consumed all available segments. EVOLVE attempts to stay within the bounds set by* nsmax *in* `parameters.in` *when resizing, but enough segments must be available to accurately describe the surface. Increase* `nsmax` *in* `parameters.in.`

Q: What are some common non-fatal errors? (Why do the results look so strange?)

A: Some common non-fatal errors are:

- *Time steps are too large. Halve* timax *or* target, *or double* nstep.
- nins *is too small to accurately represent quickly expanding corners. Increase* nins.
- *Etch is far too anisotropic due to failure to consume an activating species such as ions. If the overall sticking factor listed in the header for flat-wafer data is note exactly 1, then some fraction of the species is being re-emitted with a cosine distribution.*
- *The number of increments in zenith integrations (*ntheta*) is too small to represent a tight flux distribution.*
- *Failure to converge/solver "blows up": read the section on* relax *and* tol.

SOME HELPFUL HINTS

- *Make sure your reaction parameters are consistent with the type of reaction you are using.*
- *Make sure the diffusivity parameters are consistent with the type of diffusion you are using.*

- *Make sure your operating conditions are entered in with the correct units.*
- *The surface coverages used in* irtype *2 are not surface fractions, but absolute coverages. Make sure your rate coefficients reflect this.*

Tips for running REFLOW simulations:

- *Define all melted film parameters. Input them into reflow.in.*
- *Alter total process time (timax), number of time steps (nstep), and plot frequency (nploto) to complete the simulation with good spacing between profiles.*
- *Check to make sure that (spaces) defines a proper mesh density for resolution, while minimizing simulation space.*
- *Change one parameter at a time to investigate the effects on simulations.*

The History of EVOLVE

EVOLVE was started by Tim Cale in 1990 at Motorola. Motorolans who supported the effort in the early days include Ed Hall, Tom Zirkle and Len Borucki. Over the years, SRC, NSF, the DoD and a number of companies have supported the development of EVOLVE. At the SRC, Bill Lynch, Bill Holton, Mike Witty, Bill Atkins and Jim Hutchby have helped in a variety of ways.

The various versions of the low pressure module have been written by Tim Cale. That code forms the basis for much of the code in the high pressure and thin film flow modules.

The first versions of the high pressure module and thin film flow module were written by Hang Liao for his Ph.D. thesis at Arizona State University.

Zongwu Tang was the first to integrate these three modules. That work was completed in 1997. The current, more integrated version was written by Tim Cale, Max Bloomfield and Sofiane Soukane.'

The moving algorithm has taken quite a bit of effort, with input from a number of people. They include: Ganesh Rajagopalan, Max Bloomfield, Sofiane Soukane, and Vadali Mahadev.

Over the years, a number of colleagues have helped guide the development of EVOLVE. In addition to those listed above, colleagues who have contributed code, testing, and/or significant input include:

- Tushar Merchant at Motorola
- Dan Coronell at Motorola
- Andrew Labun at DEC and Intel
- Eric Egan at Motorola
- Anthony Toprac at SEMATECH
- Valery Sukharev at LSI

- Mike Coltrin at Sandia National Labs
- Mary Hudson at Sandia National Labs
- Fletcher Jones at IBM

In addition to these contributors, there have been a large number of people who I have worked with over the years to define models.